NTNU
Kunnskap for en bedre verden

Department of Mathematical Sciences

TMA4285 - Time Series

Forecasting a cryptocurrency - Part 2

Authors:

Martin Lie, Camille Loisel, Frederick Nilsen

October, 2021

Contents

List of Figures

List of Tables

1 Abstract

This submission is to be intended as a general commentary to the first project. We now introduce models that are more applicable to financial time series, mainly GARCH and some of its variations. We wish to determine if these methods produce a better forecast than the original ARIMA-modeling and discuss the results of these predictions. As suspected in the first project, a distribution with heavy tails suited our dataset better and hence our predictions were significantly improved.

2 Introduction

In the first part of the project [3], we used traditional ARIMA-models in order to attempt to understand and forecast future stock values of a certain cryptocurrency, Litecoin. Modeling the financial time series with an ARIMA-model deemed difficult since it couldn't account for the stock's volatility and since the general assumption of stationarity was not met.

In addition, we found our residuals to be approximately Cauchy-distributed [3]. It therefore becomes necessary to add on the effect of heavy-tails in order to model our data more appropriately as the model was significantly impacted by great deviations in both variance and mean. Thus, we will look into the following GARCH models in this commentary that we find the most relevant for our data:

- Standard GARCH (Normality assumption)
- ARMA-GARCH (ARMA-process with GARCH noise)
- T-GARCH (student t assumption with ν degrees of freedom)
- E-GARCH (Allowing both positive and negative values for the process)

3 Theory

3.1 Standard ARCH and GARCH model

While AR(I)MA models are well suited for dealing with data sets of finite and constant variance, these assumptions generally break down for financial data sets. As we observed in [3], the *volatility* varied for different time periods. A more advanced model is needed to accommodate for these changes and we introduce *(G)ARCH*. The ARCH model is an acronym for *Autoregressive Conditional Heteroscedasticity*, defined below [1, p. 197].

Definition 3.1 (ARCH process)**.** $\{Z_t\}$ is assumed to be an ARCH process, if:

$$Z_t = h_t^{\frac{1}{2}} e_t,$$

where

$$\begin{aligned}
&\{e_t\} \sim \mathcal{N}(0,1) \text{ (iid)},\\
&\alpha_0 > 0, \quad \alpha_j \geq 0, \quad j = 1, \ldots, p\\
&e_t \text{ and } Z_{t-1}, \ldots, Z_{t-p} \text{ are independent for all } t.
\end{aligned}$$

The main takeaway is that ARCH models can be applied to heteroscedastic data sets, e.g. financial time series with a changing variance. Z_t is conditionally Gaussian, given past values $\{Z_s, s = t-1, \ldots, s = t-p\}$. In general terms, Z_t is a strictly stationary process for $\alpha_1 < 1$. However, for financial time series, these assumptions are generally relaxed in order to obtain overall better fits. In these cases, we define $Z_t = log(\frac{P_t}{P_{t-1}})$, where P_t is the price of a stock by the end of day t. We also define the generalized version of ARCH known as a GARCH model [1, p. 200]

Definition 3.2 (GARCH process)**.** $\{Z_t\}$ is said to be a GARCH(p,q) process, if it is stationary, and

$$Z_t = h_t^{\frac{1}{2}} e_t, \tag{1}$$

where

$$\begin{aligned} \{e_t\} &\sim \mathcal{N}(0,1), \text{ (iid) }, \\ h_t &= \alpha_0 + \alpha_1 Z_{t-1}^2 + \cdots + \alpha_p Z_{t-p}^2 + \beta_1 h_{t-1} + \cdots + \beta_q h_{t-q} \\ &= \alpha_0 + \alpha(B) Z_t^2 + \beta(B) h_t, \end{aligned}$$

where B is the backwards operator in the last equation.

The GARCH model is indeed very general, and there exists several variations of the model that excel at different measurements. We present now the GARCH variations used in this project. When testing for different models of financial series, we look for the following properties in our series:

1. The underlying marginal distributions have heavy tails,
2. The volatility is persistence,
3. The sum of returns Z_t, $S \sim$ (asymptotically) normal for $n \longrightarrow \infty$,
4. An asymmetry related to psotive and negative returns is present,
5. Long-term dependence in the volatility is present.

One of the variations is in fact not a GARCH-model, but an ARMA-GARCH model. This is an ARMA-process as defined in [3], with the modification that the white noise parameter is replaced with a GARCH-model. Thus, the noise component can more accurately account for volatility across the time series while still using a modified ARMA-model. This model, T-GARCH (mentioned below) and sGARCH accounts for the three first properties listed above.

Another model we would like to examine, is the T-GARCH model. This model is a GARCH model whose $\{e_t\}$ is not iid normally distributed, but rather student-t distributed with ν degrees of freedom. This is especially interesting to try to fit the data with, since we from [3] observed a heavy tail distribution being involved in our estimates.

The final GARCH variation to be examined is a E-GARCH model. This one also includes the last two properties mentioned. Assuming that our process follows equation (1), we define the set $\{l_t = ln(h_t)\}$ to be a collection of both strong and weakly solutions solutions of the equations

$$l_t = c + \alpha_1 g(e_{t-1}) + \gamma_1 l_{t-1}, \tag{2}$$

where $c, \alpha_1 \in \mathbf{R}$, $|\gamma_1| < 1$, and

$$g(e_t) = \begin{cases} (1+\lambda)e_t - \lambda E[|e_t|] & \text{if } e_t \geq 0 \\ (1-\lambda)e_t - \lambda E[|e_t|] & \text{if } e_t < 0. \end{cases} \tag{3}$$

We therefore assume e_t has a symmetric distribution with $e_t \overset{\text{in distribution}}{=} -e_t$. The advantage of our model comes from the general form of $g(e_t)$, where the slope changes depending on the interval e_t lies within. In practice, our model therefore responds asymmetrical to changes within given values of e_t. Symmetry is only achieved for $\lambda = 0$. [1] generally suggest assuming that e_t follows *generalized error distribution* (GED), where $\epsilon = (\frac{\Gamma(\frac{1}{\upsilon})2^{-\frac{2}{\upsilon}}}{\Gamma(\frac{3}{\upsilon})})^{\frac{1}{2}}$ and $\upsilon > 0$ is the number of degrees of freedom. We define the density distribution of $X \sim GED$ as

$$f(x) = \frac{\upsilon \exp(-\frac{1}{2}|\frac{x}{\epsilon}|^{\upsilon})}{\epsilon 2^{1+\frac{1}{\upsilon}}\Gamma(\frac{1}{\upsilon})}. \tag{4}$$

υ generally determines how heavy the tails of the distribution is and it increases as υ decreases.

Now let $\alpha(B) = \sum_{i=1}^{m} \alpha_i B^i$ and $\beta(B) = \sum_{i=1}^{n} \beta_i B^i$, and we define a general **E-GARCH(m,n)**-process as

$$l_t = c + \alpha(B)g(e_t) + \beta(B)l_t. \tag{5}$$

3.2 Nyblom stability and Sign-Bias Tests

Useful analytical tools for testing the overall fit of an GARCH model is *Nyblom stability* and *Sign Bias* tests. Nyblom stability tests for structural changes within the estimated regression coefficients. H_0 is set that these coefficients remain constant, while H_1 is that their vary (if τ_0^2 is the variance we have that $H_0 : \tau_0^2 = 0$ vs. $H_1 : \tau_0^2 > 0$) [2]. The sign bias test consists of three tests where each of them have the following properties:

- *Sign Bias Test*: We set a dummy variable S_- to 1 when $e_t < 0$, and test for the effects the positive and negative shocks has on the volatility not predicted by the model.
- *Positive Sign Bias Test*: Using the same dummy variable, we simply test for the effect of small and large negative shocks.
- *Negative Sign Bias Test*: We define a new dummy variable by $S_+ = 1 - S_-$ and test for the effect of small and large positive effects.

Generally for the sign test, we have that $H_0 : \sum S_- = 0$ vs. $H_1 : \sum S_- > 0$ or $H_0 : \sum S_+ = 0$ vs. $H_1 : \sum S_+ > 0$ depending on the test.

4 Results

We produce a similar strategy for fitting and forecasting with all GARCH variations, consisting of the following

1. Load and pre-process Litecoin stock data by removing NA-entries and defining $Z_t = log(\frac{P_t}{P_{t-1}})$ for the financial setup. We also implement an ARIMA-model to our processed model to find the best mean model.
2. Transform the stock data with log-transform and 1-step difference.
3. Brute-force check for what combination of p and q that leads to smallest Akaike and Bayes information criteria and select the optimal choices based on these estimates.
4. Fit and forecast with the fitted model of optimal p and q.

4.1 Model selection

Initial forecasts with no transformation of the time series data yielded in mediocre results. As such, we will only be presenting the forecast with a log-difference transformation of the data, i.e. we instead model with the logarithm of the data and using one-step differencing. Using the `auto.arima`-function, we find that the best mean model is ARMA$(1, 1)$ with AIC$= -5046.21$ and

BIC= -5024.12. This is the same model that had the overall best fit in the first part of the project [3]. We summarize the results of the information criteria for our four models in Table 1. In addition, we show the relevant coefficients for the three modified models in Tables 2, 3 and 4, respectively.

For sGARCH, We find that $(p, q) = (10, 7)$ produces the lowest AIC $= -2.87$, and the optimal BIC $= -2.83$ is for $(p, q) = (1, 3)$. Although the AIC-choice complicates our model, we chose to model with $(10, 7)$, since it yielded more promising results, and allowed for more complex model since the sGARCH is not as complex as its variations on its own. For the other models, we get the same optimal parameters for with both information criteria tests. For T-GARCH, AIC= -3.16 and BIC $= -3.14$. For E-GARCH, we find that the best fit is of order $(1, 1)$ with AIC= -3.14 and BIC= -3.12, testing for values up to $p = q = 15$.

4.2 Residual fit

Using the function `ugarchfit` (from the R-package `rugarch`), we are able to produce plots of the autocovariance function (ACF) and density of the standardized residuals by means of determining the model fit graphically. The ACF is found in Figure 3, the ACF of the squared residuals are in Figure 4 and density plots in Figure 5. We are also able to produce the QQ plots of the models as shown in Figure 6 helping us determine normality of the residuals. A more comprehensive discussion of the model fits are available in Section 5.2. Finally, for the E-GARCH model we include the *News Impact Curve*, which is an estimated version of equation (3), given how our model responds differently to positive and negative values of e_t.

We also refer to the forecasts of the Litecoin stock price for the next $n = 27$ time steps (days) ahead of the sampled data. In the forecasts, the x-axis resembles indices, i.e. number of days passed since September 27th, 2016, whereas the y-axis resembles the stock price in terms of US dollars. These forecasts are shown in Figures 1a, 1b, 1c and 1a. We also show the forecast in relation to the previous values in full in Figure 2.

The following tables summaries each of our models. Using the `summary`-function in R we find the estimated values and their significance while we also test for *Nyblom stability* and the *Sign Bias*. The *Ljung-Box* test confirms no serial correlations between the standardized residuals for the models.

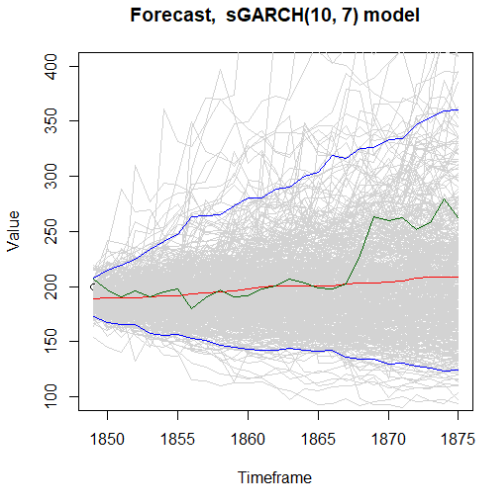
5 Discussion

5.1 Forecasting results and comparison

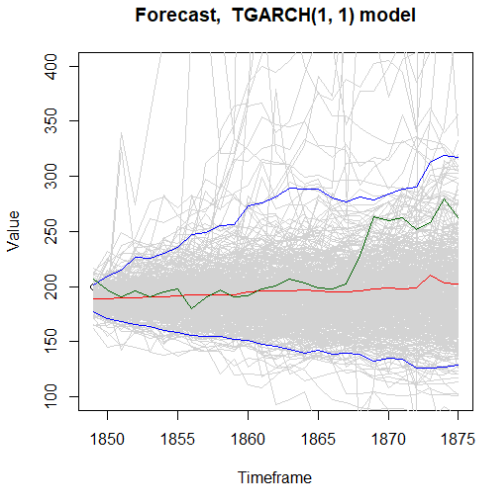
Table 8 showcases the final results. We note that in terms of the prediction error (comparing the forecasted new values to the true observations), E-GARCH is the one that manages to significantly perform better than the others. However, this model also contains the largest prediction interval, though is remains more skewed towards the upper ends. Given that our final value for the data set was 188.61, this model predicts that the stocks is more likely to increase significantly than decreasing which turned out to be the real case. Using the GED-distribution for ARMA-GARCH and T-GARCH gives of the smallest intervals. We do however note that neither of these predic-

Model	AIC parameters	BIC parameters
sGARCH	$(10, 7)$	$(1, 3)$
T-GARCH	$(1, 1)$	$(1, 1)$
ARMA-GARCH	$(1, 1)$	$(0, 0)$
E-GARCH	$(1, 1)$	$(1, 1)$

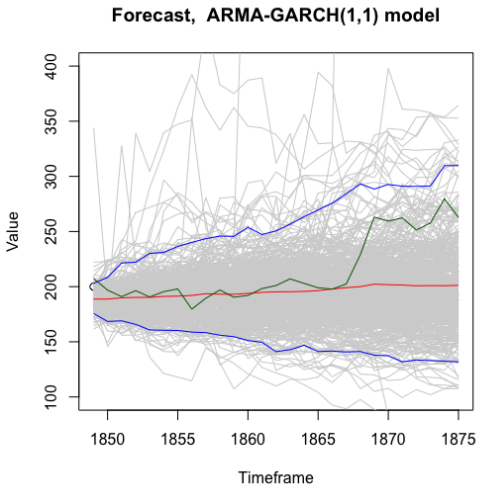
Table 1: Optimal AIC and BIC values for the different models, fitted on the log-transformed one-step differenced stock values.



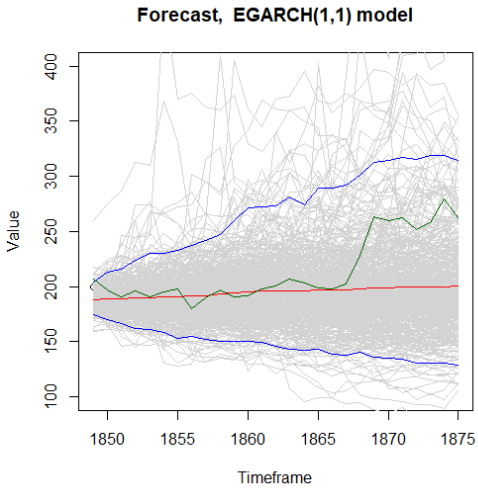
(a) sGARCH forecast



(b) T-GARCH forecast



(c) ARMA-GARCH forecast



(d) EGARCH forecast

Figure 1: Forecasting results of the 4 models, forecasting $n = 27$ days ahead of last sampling date. The black lines are the sampled data, the green line are the actual test observations omitted form the model fit, and the grey ones are resulting predictions from each bootstrap. Finally, the red line signifies the mean forecast (over each grey line), and the blue lines indicate the 97.5 % confidence interval.

Type	μ	ϕ_1	θ_1	ω	α_1	β_1
Value	-3×10^{-6}	0.028596	−0.122575	9.1×10^{-5}	0.117693	0.881307
Standard error	0.000774	0.38548	0.382441	3.9×10^{-5}	0.022138	0.023585
t-value	−0.003682	0.074182	−0.320506	2.306896	5.316457	37.367786
p-value	0.997062	0.940866	0.748585	0.021061	0.0	0.0
Nyblom stability	0.2434	0.3261	0.3228	1.4513	0.1988	0.3571

Table 2: Summary of the T-GARCH coefficients

Type	μ	ϕ_1	θ_1	ω	α_1	β_1
Value	-3×10^{-6}	0.181285	-0.293484	9.5×10^{-5}	0.102813	0.879809
Standard error	0.000774	0.011797	0.012025	3.3×10^{-5}	0.021691	0.022409
t-value	-0.003682	15.3671	-24.4056	2.8756	4.7399	39.2611
p-value	0.997062	0.0	0.0	0.004032	2×10^{-6}	0.0
Nyblom stability	0.2434	0.74048	0.75342	0.32597	0.06844	0.14636

Table 3: Summary of the ARMA-GARCH coefficients

Type	μ	ϕ_1	θ_1	ω	α_1	β_1	γ_1	λ
Value	0.000035	-0.006879	-0122107	-0.179890	0.022894	0.969256	0.194412	0.920790
Standard Error	0.000044	0.002666	0.009661	0.039284	0.016724	0.006668	0.026874	0.034684
t-value	0.79273	-2.58021	-12.63905	-4.57718	1.36897	145.36703	7.23700	26.54780
p-value	0.427935	0.009874	0	0	0.171010	0	0	0
Nyblom stability	1.08034	0.36644	0.40781	0.06938	0.19770	0.07162	0.32572	0.88506

Table 4: Summary of the E-GARCH coefficients

Values	t-value	Sign probability
Sign Bias	0.14079	0.8880
Negative Sign Bias	0.32178	0.7477
Positive Sign Bias	0.08612	0.9314
Joint Effect	0.22414	0.9736

Table 5: Sign - Bias Test for E-GARCH

Values	t-value	Sign probability
Sign Bias	0.2276	0.8200
Negative Sign Bias	0.4350	0.6636
Positive Sign Bias	0.2496	0.8029
Joint Effect	0.4366	0.9326

Table 6: Sign - Bias Test for ARMA-GARCH

Values	t-value	Sign probability
Sign Bias	0.2489	0.8034
Negative Sign Bias	0.4637	0.6429
Positive Sign Bias	0.3276	0.7432
Joint Effect	0.5102	0.9166

Table 7: Sign - Bias Test for T-GARCH

Model	Prediction Error	97.5% Confidence interval	Range/Size of interval
ARMA-GARCH	20.73	(131.76, 309.76)	178
sGARCH	20.52	(113.67, 357.02)	243.35
T-GARCH	22.03	(126.25, 333.93)	207.68
E-GARCH	18.5	(121.49, 380.96)	259.47

Table 8: Summary of forecast errors, 97.5% confidence interval and the range of the corresponding interval for all models. All forecasts preformed with 500 bootstrap predicitons.

tions increases at the same magnitude as with E-GARCH. While sGARCH contains a relatively symmetric form of the confidence intervals, these intervals becomes increasingly asymmetrical with T-GARCH, ARMA-GARCH and E-GARCH, where the latter has the most skewed form.

5.2 Model and residual fit

From Figure 6, we note that the residuals of sGARCH has the poorest fit, due to the nature the of heavy-tail distributed data, explaining why the other three perform better. In addition, we note from our distribution plot for the residuals in Figure 5a, that the normal fit corresponds poorly for sGARCH. Figure 3a also signifies the presence of bigger serial correlation than the other three. In fact, we get a relatively slow decay of the ACF for all models as seen in Figure 3. Looking at the squared standardized residuals however in Figure 4, we observe relatively rapid decay, especially in 4b and 4c.

Comparing the latter three to each other, no significant improvements are noted, but including the GED-distribution in our models does seem to suit the data a little better when comparing Figures 6b to 6d, and similarly for 5b compared to 5d. However, the figures remain very much the same in Figures 3 and 4 for the three modified models. The main difference between E-GARCH, T-GARCH and ARMA-GARCH is how future events impact our predictions. From Figure 7, we have estimated the value of λ from equation (3). Here, future positive errors or news impacts the value of the stock significantly more than negative ones. This again is reflected in the prediction, where E-GARCH allows for a bigger and more profitable prediction given that it far more likely that our stocks end up growing in value. Tables (3) and (4) contains small differences, but do note that T-GARCH have fewer significant values, while the latter two has few non-significant parameters. None of the tables shows any significant results for the Nyblom stability and the same applies for tables (5), (6) and (7) where no significant changes on both positive and negative values in regards to the volatility was noted. This confirms that our models hence had an overall good fit to the set.

5.3 Comparison to ARIMA-model from first project

It is especially relevant to compare the results from the GARCH-forecasts with the results from [3]. There are however some differences that has to be taken into account. The first issue is that the models are both created and forecasted with different amount of values. The models presented in this report has 1847 observations used in model fit, and 27 for testing, whereas in [3] we have 1827 and 21 for fitting and testing, respectively. As a result, the models in [3] forecast on a completely different date than the models presented here, introducing potential comparison errors as one forecast range may be less volatile than another. Another anomaly worth mentioning is that [3] calculated the 95% confidence interval of the forecast and not 97.5% as done here.

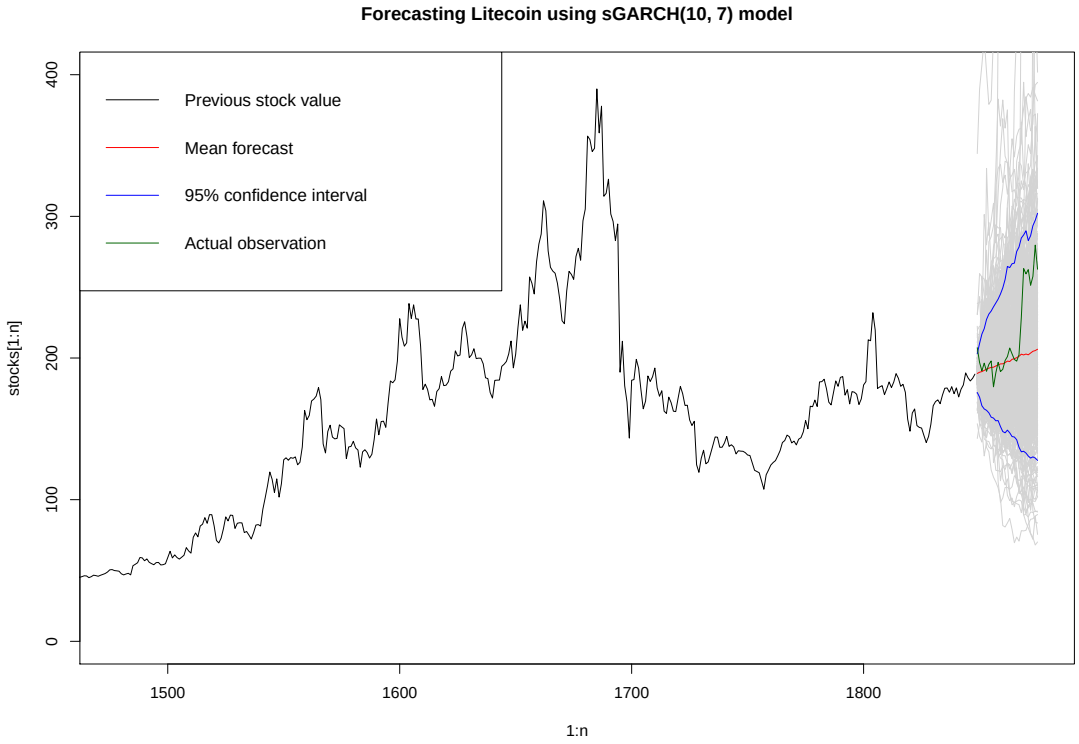
Keeping these issues in mind, we still observe a significant decrease in the mean error in Section 4.2 compared to [3]. By studying the true observations, we also observe that the test data for the GARCH models have more sudden jumps than the test data for the ARIMA models. This indicates that the difference in mean error could potentially be even larger had the models been tested on the same time spans.

6 Conclusion

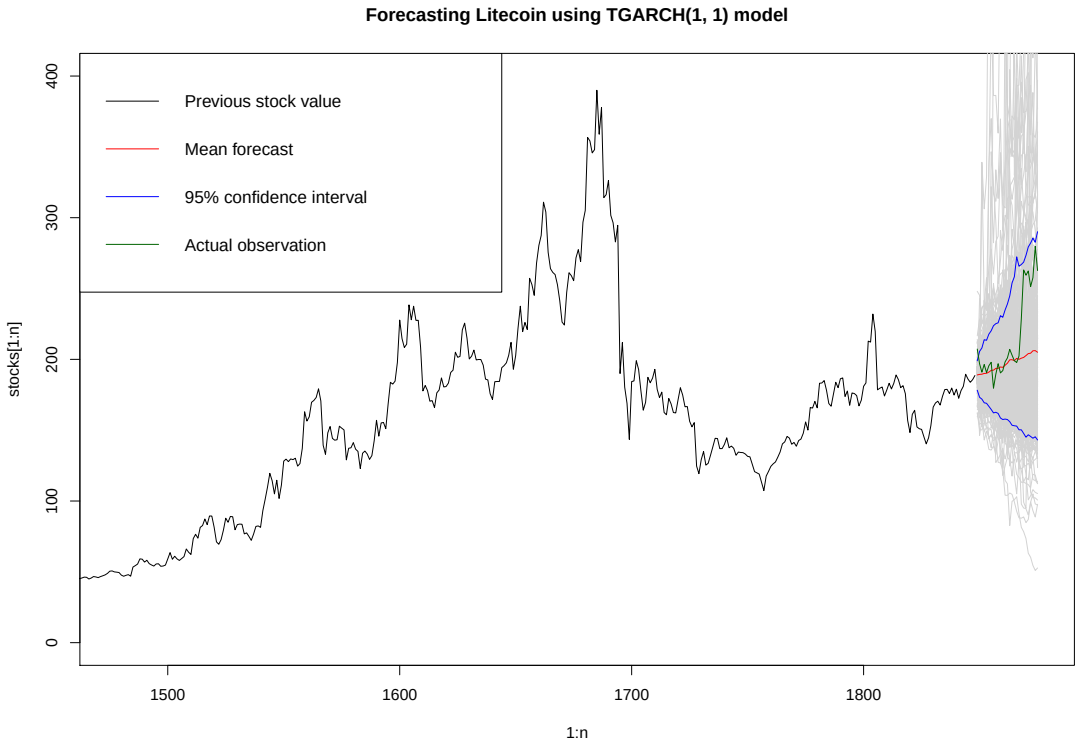
While [3] aimed to forecast a volatile cryptocurrency using only AR(I)MA models, this commentary has attempted to model the same dataset with more complex and appropriate models for financial series. The GARCH-models, although forecasting similarly, are able to predict with significant less error than the ARIMA models. They also managed to generally capture the more likely outcomes in in that the stocks was far more likely to increase in value. Assuming one were considering investing in these stocks, it would be profitable buying stocks with one month to maturity.

Appendix

A Additional Figures and Tables

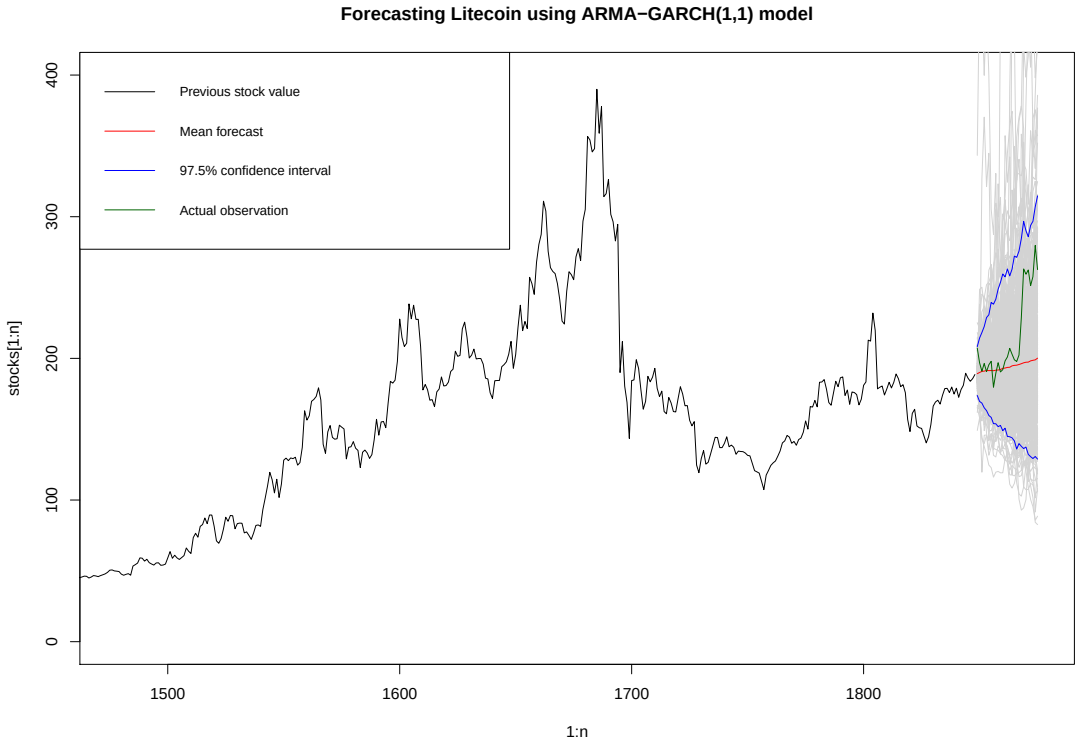


(a) sGARCH forecast

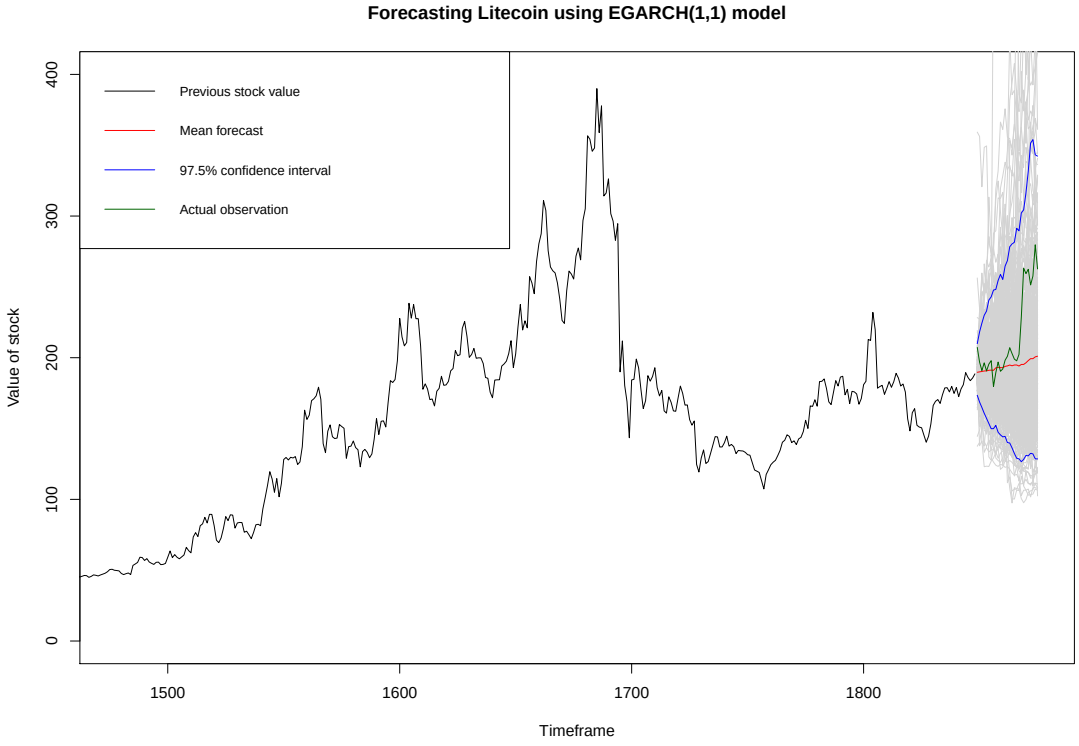


(b) T-GARCH forecast

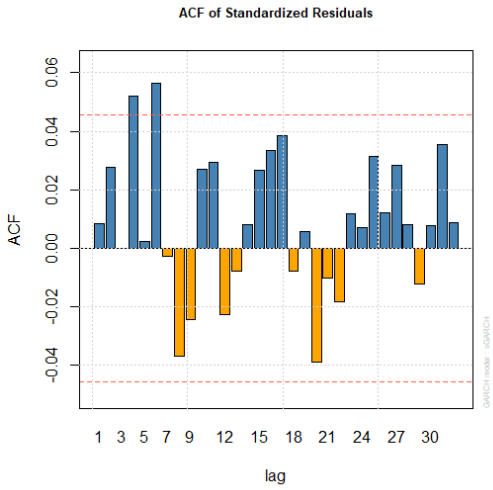
Figure 2: Full forecasting results for the four models



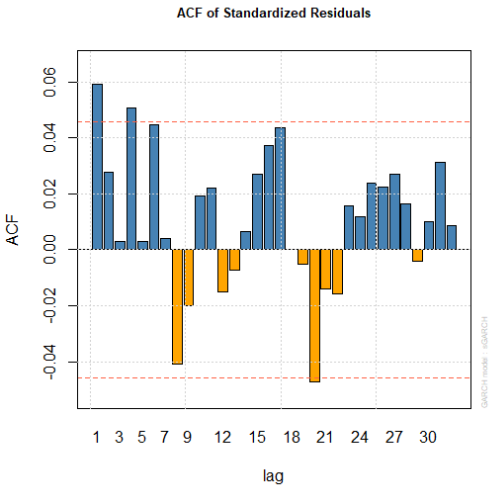
(c) ARMA-GARCH forecast



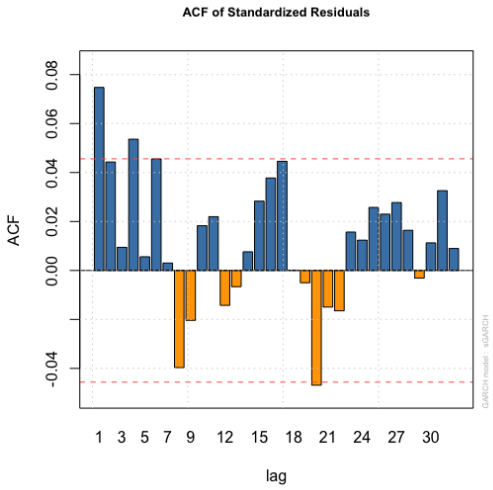
(d) EGARCH forecast



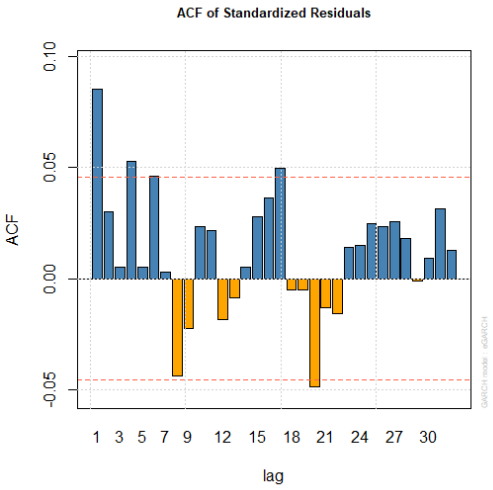
(a) standard GARCH



(b) T-GARCH

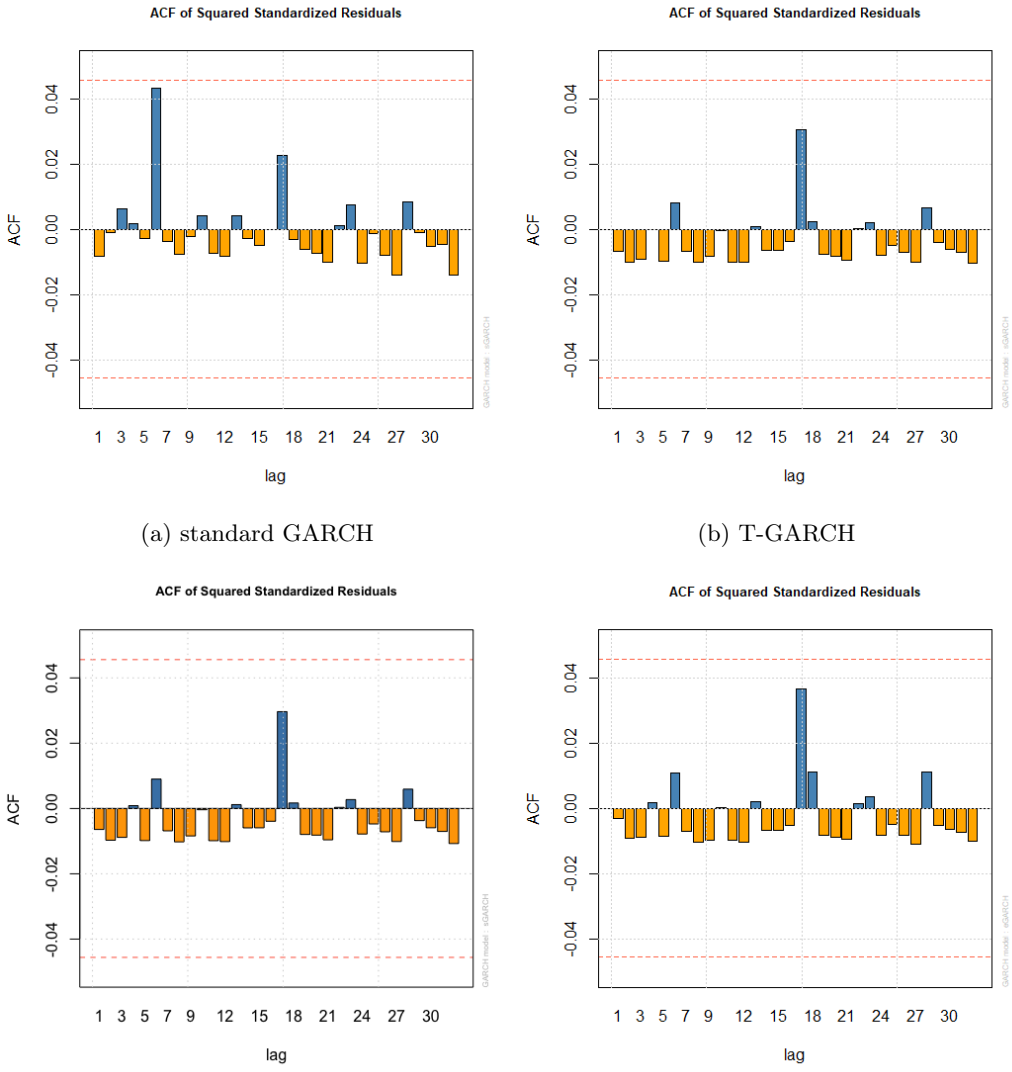


(c) ARMA-GARCH



(d) E-GARCH

Figure 3: Autocovariance of standardized residuals over lag for the four models.



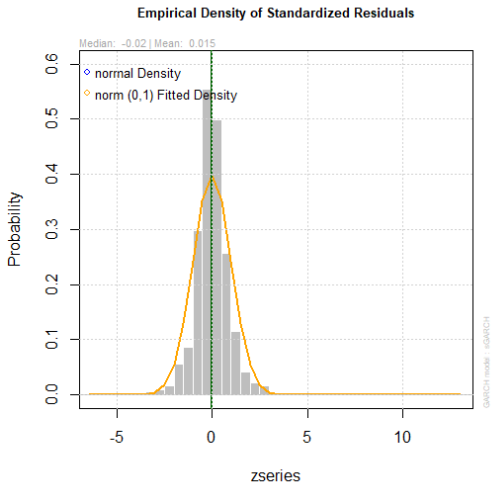
(a) standard GARCH

(b) T-GARCH

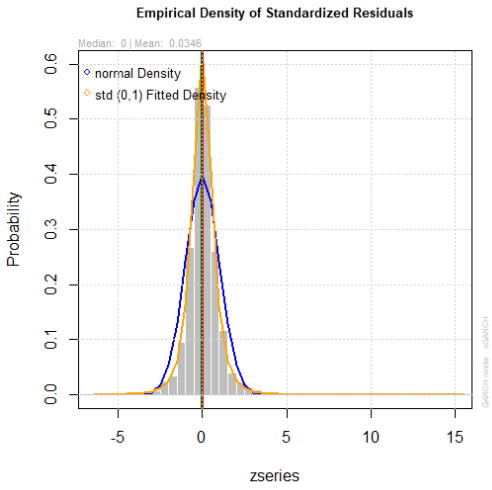
(c) ARMA-GARCH

(d) E-GARCH

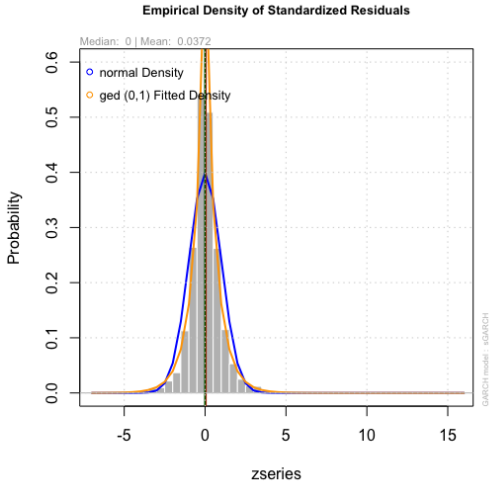
Figure 4: Autocovariance of *squared* standardized residuals over lag for the four models.



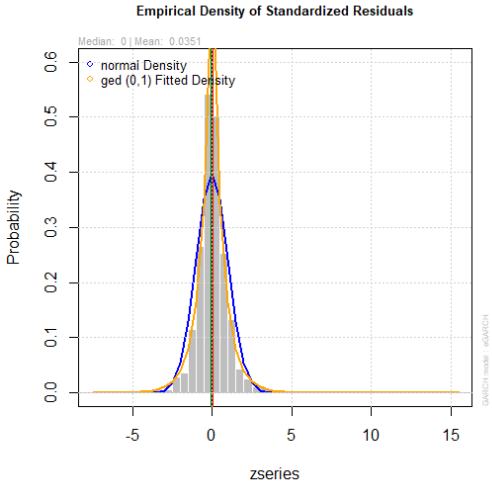
(a) standard GARCH



(b) T-GARCH

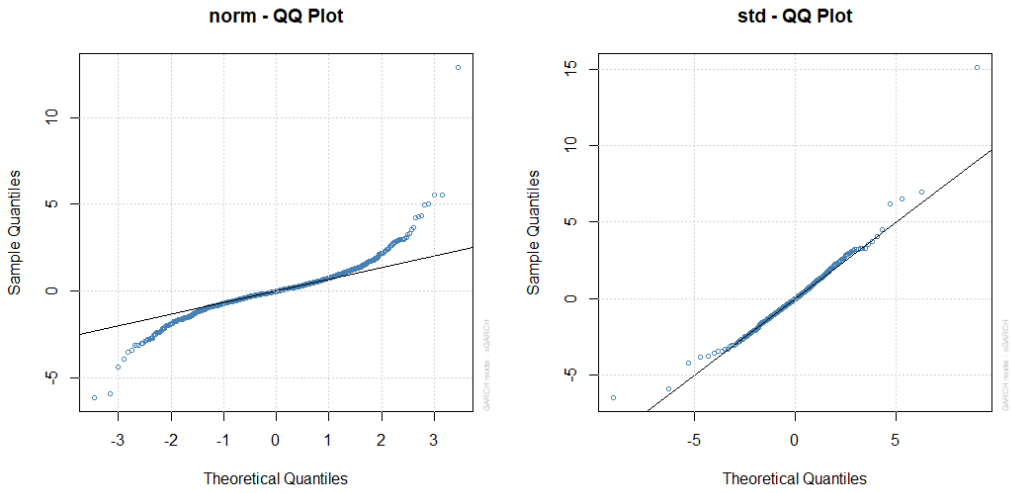


(c) ARMA-GARCH

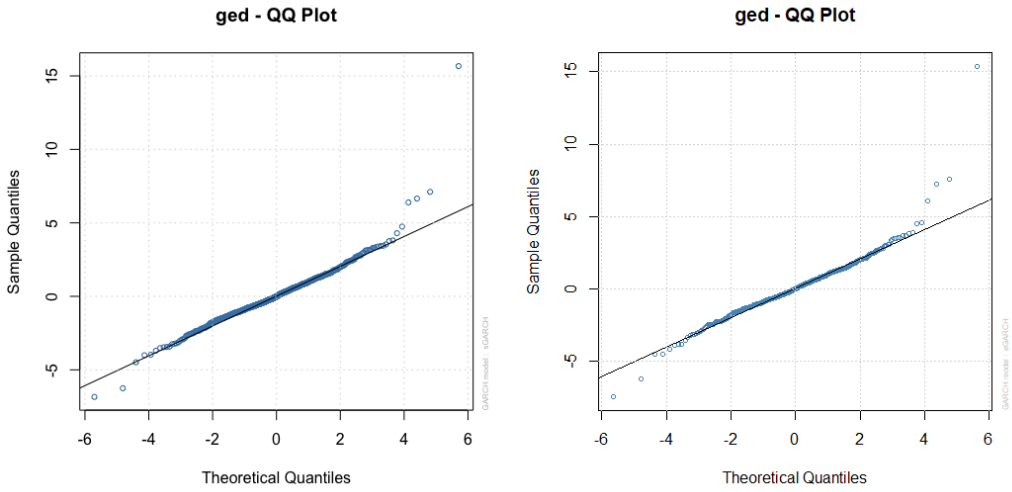


(d) E-GARCH

Figure 5: Density of standardized residuals for the four models.



(a) standard GARCH (b) T-GARCH



(c) ARMA-GARCH (d) E-GARCH

Figure 6: QQ plot for the four models.

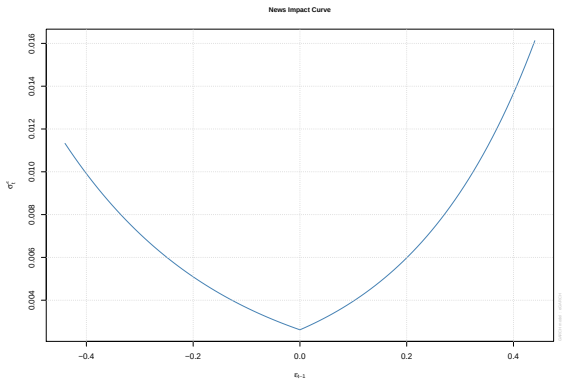


Figure 7: The News Event impact of future and past GED-error terms

B R code

Preprocessing the data set:

```
# Preprocessing and data cleaning
data = read_excel('CBLTCUSD_updated.xls')
data <- na.omit(data)
stocks <- data$CBLTCUSD # Remove dates and convert to numeric
plot(data)

log.stocks <- log(stocks)
diff.log.stocks <- diff(log.stocks)

# Testing observations
trueobs_data <- read_excel('CBLTCUSD_update_2.xls')
trueobs <- trueobs_data$CBLTCUSD
```

Helper functions:

```

# Naive/Brute force p and q selection
naive_selection <- function(data, pqmax, distrmod){
    # Helper to select p and q based on lowest AIC/BIC
    # :param data:      should be stocks or the log-diff transformed stock data
    # :param pqmax:     range of combinations to examine, e.g. pqmax=3 will check
    #                   every combination between (0,0) and (3,3)
    # :param distrmod:  Distribution model, e.g. "std" or "norm"
    # :return:          Array of lowest AIC/BIC (p and q printed to console)
    info.res <- vector(mode="list", length=4)
    names(info.res) <- c("p", "q", "aic", "bic")
    pb_iter <- (pqmax+1)^2 # Number of iterations
    pb_count <- 0
    pb <- txtProgressBar(min=pb_count, max=pb_iter, style=3, width=50, char="=")

    for(p in 0:pqmax){
        for(q in 0:pqmax){
            # Create and fit model for each (p,q)
            tmp_model <- ugarchfit(
                spec=ugarchspec(
                    variance.model = list(garchOrder=c(p,q)),
                    #mean.model = list(armaOrder=c(0,1), include.mean = FALSE),
                    distribution.model = distrmod
                ),
                data=data
            )
            # log(log(nObs)) may produce NaNs in some cases, so
            # we skip iterations with invalid AIC/BIC
            skip_iter <- FALSE
            tryCatch(
                calc <- infocriteria(tmp_model),
                error = function(e){ skip_iter <<- TRUE}
            )
            if(skip_iter) { next }

            info.res$p <- append(info.res$p, p)
            info.res$q <- append(info.res$q, q)

            # Extract first 2 results, since we just want Akaike and Bayes
            info.res$aic <- append(info.res$aic, calc[1])
            info.res$bic <- append(info.res$bic, calc[2])
            # Update progress bar
            pb_count <- pb_count + 1
            setTxtProgressBar(pb, pb_count)
        }
    }
    print("Completed naive p, q test")
    info.optimal <- vector(mode="list", length=4)
    names(info.optimal) <- c("aic_val", "bic_val", "aic_param", "bic_param")

    best_aic_index = match(min(info.res$aic), info.res$aic)
    best_bic_index = match(min(info.res$bic), info.res$bic)

```

```
    info.optimal$aic_val <- append(info.optimal$aic_val, min(info.res$aic))
    info.optimal$bic_val <- append(info.optimal$bic_val, min(info.res$bic))

    info.optimal$aic_param <- append(
        info.optimal$aic_param,
        c(info.res$p[best_aic_index], info.res$q[best_aic_index])
        )
    info.optimal$bic_param <- append(
        info.optimal$bic_param,
        c(info.res$p[best_bic_index], info.res$q[best_bic_index])
        )

    cat("Best AIC for (p,q) = (", info.optimal$aic_param[1],
        ",", info.optimal$aic_param[2], ")", sep="")
    cat("Best BIC for (p,q) = (", info.optimal$bic_param[1],
        ",", info.optimal$bic_param[2], ")", sep="")
    return(info.optimal)
}

best_choice_std <- naive_selection(diff.log.stocks, 4, "std")
best_choice_norm <- naive_selection(diff.log.stocks, 10, "norm")

# Backtransform log-diff ts
init = stocks[1]
difflogdata = diff.log.stocks
backtransform <- function(x, difflogdata, init){
    n <- length(difflogdata)
    ts.all <- c(difflogdata,x)
    cs <- cumsum(ts.all)
    newts <- exp(cs[(n+1):length(ts.all)])*init
    return(newts)
}

# Plot forecasted values
plot_forecast <- function(data, trueobs, n.ahead, ylim, forecasted_model, modeltxt)
    {
    # Generalized plotter function to plot all forecasts similarly
    # data = time series used in modeling
    # trueobs = testing data not used in modeling
    # n.ahead = number of time steps forecasted for
    # forecasted_model = Bootrap-forecasted and backtransformed data
    # modeltxt = string representation of model used

    # Plot forecast of backtransformed data
    n <- length(data)
    plot(1:n,stocks[1:n],type="l",xlim=c(as.integer(n/1.25),n+n.ahead),ylim=c(0,
    ylim),
         main= paste("Forecasting Litecoin using", modeltxt, "model"))
    for(i in 1:nrow(forecasted_model)){
        lines(seq(n+1,n+n.ahead),forecasted_model[i,],col="lightgrey")
    }
    lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, mean),col="red")
    lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.025),col="
    blue")
    lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.975),col="
    blue")
    lines(seq(n+1,n+n.ahead), trueobs,col="darkgreen")
    legend("topleft", legend=c(
        "Previous stock value",
        "Mean forecast",
        "97.5% confidence interval",
        "Actual observation"),
        col=c("black", "red", "blue", "darkgreen"),
        lty=1, cex=1.05)
}

plot_forecast_minimized <- function(data, trueobs, n.ahead, ylim, forecasted_model,
     modeltxt){
    # Generalized plotter function to plot all forecasts similarly
    # data = time series used in modeling
    # trueobs = testing data not used in modeling
    # n.ahead = number of time steps forecasted for
    # forecasted_model = Bootrap-forecasted and backtransformed data
```

```
    # modeltxt = string representation of model used

    # Plot forecast of backtransformed data
    n <- length(data)
    plot(n+1, 200, xlim=c(n+1,n+n.ahead),ylim=c(100,ylim),
         xlab="Timeframe", ylab="Value",
         main= paste("Forecast, ", modeltxt, "model"))
    for(i in 1:nrow(forecasted_model)){
        lines(seq(n+1,n+n.ahead),forecasted_model[i,],col="lightgrey")
    }
    lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, mean),col="red")
    lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.025),col="
    blue")
    lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.975),col="
    blue")
    lines(seq(n+1,n+n.ahead), trueobs,col="darkgreen")
}


numeric_quantile <- function(n.ahead, forecasted_model){
    # Helper to get numeric values of confidance interval to the n.ahead-th
    prediction
    all_lower <- apply(forecasted_model, 2, quantile,probs=0.025)
    all_upper <- apply(forecasted_model, 2, quantile,probs=0.975)
    conf_interval <- c(all_lower[n.ahead], all_upper[n.ahead], abs(all_lower[n.
    ahead]-all_upper[n.ahead]))
    return(conf_interval)
}
mean_error <- function(forecasted_model, trueobs){
    # Helper to get mean absolute error between mean forecast and true observations
    mean_preds <- apply(forecasted_model, 2, mean) # goes from [len(dataset), len(
    dataset) + n.ahead]
    err <- abs(mean_preds - trueobs)
    return(mean(err))
}
```

Standard GARCH model:

```
# >>>>> Model 0 : sGARCH
best_choice_norm <- naive_selection(diff.log.stocks, 10, "norm")
# Fit with the p and q found from function above
# Use AIC parameters since they produce best forecast results
model0.logdiff <- ugarchfit(
    spec=ugarchspec(
        variance.model = list(garchOrder=best_choice_norm$aic_param),
        mean.model = list(armaOrder=c(0,1)),
        distribution.model = "norm"
    ),
    data=diff.log.stocks
)


for(plotNo in 8:11){
    plot(model0.logdiff, which=plotNo)
}

model0.logdiff.forecast <- ugarchboot(model0.logdiff, n.ahead=27,
                                      method="Partial", n.bootpred =500)
model0.logdiff.forecast <- t(
    apply(model0.logdiff.forecast @ fseries,
          1,
          backtransform,
          difflogdata=difflogdata,
          init=init)
)
plot_forecast(data=stocks, trueobs=trueobs, n.ahead=27, ylim=400,
              forecasted_model = model0.logdiff.forecast, modeltxt=paste(
                  "sGARCH(", best_choice_norm$aic_param[1], ", ",
                  best_choice_norm$aic_param[2], ")", sep=""
                  )
              )

```

```
35
36 # <<<<< Model 0 : sGARCH
```

T-GARCH model:

```
1  # >>>>> Model 1 : T-GARCH
2
3  # Function above shows (p,q)=(1,1) to be best fit
4  model1.logdiff <- ugarchfit(
5      spec=ugarchspec(
6          variance.model = list(garchOrder=best_choice_std$aic_param),
7          distribution.model = "std"
8      ),
9      data=diff.log.stocks
10 )
11
12 # Plots indicating general model fit
13 for(plotNo in 8:11){
14     plot(model1.logdiff, which=plotNo)
15 }
16
17
18 # Predict and and backtransform
19 model1.logdiff.forecast <- ugarchboot(model1.logdiff, n.ahead=27,
20                                       method="Partial", n.bootpred =500)
21 model1.logdiff.forecast <- t(
22     apply(model1.logdiff.forecast @ fseries,
23           1,
24           backtransform,
25           difflogdata=difflogdata,
26           init=init)
27 )
28
29
30 plot_forecast_minimized(data=stocks, trueobs=trueobs, n.ahead=27, ylim=400,
31               forecasted_model=model1.logdiff.forecast, modeltxt=paste(
32                   "TGARCH(", best_choice_std$aic_param[1], ", ",
33                   best_choice_std$aic_param[2], ")", sep="")
34               )
35
36 # <<<<< Model 1 : T-GARCH
```

ARMA-GARCH model:

```
1  library(forecast)
2  # Fitting ARIMA(1,1,1) model
3  auto <- auto.arima(log.stocks,ic=c("aic"))
4  model1=arima(log.stocks,order=c(1,1,1),include.mean = FALSE)
5  summary(model1)
6
7  # ARIMA(1,1,1)-GARCH(p,q) model
8  model <-ugarchspec(variance.model = list(garchOrder=c(1,1)),mean.model = list(
       armaOrder=c(1,1),
9                                        include.mean=FALSE), distribution.model = "ged"
       )
10
11 model2 <-ugarchfit(spec=model,data=diff.log.stocks)
12 plot(model2, which="all")
13
14 # Forecasting
15 f.model2 <- ugarchforecast(model2, n.ahead = 27)
16 plot(f.model2)
17
18
19 model2.logdiff.forecast <- ugarchboot(model2, n.ahead=27,
20                                       method="Partial", n.bootpred=500)
21 plot(model2.logdiff.forecast)
22
23 model2.logdiff.forecast <- t(
24   apply(model2.logdiff.forecast @ fseries,
25         1,
26         backtransform,
```

```
        difflogdata=difflogdata,
        init=init)
)

plot_forecast(data=stocks, trueobs=trueobs, n.ahead=27, ylim=400,
              forecasted_model = model2.logdiff.forecast, modeltxt="ARMA-GARCH(1,1)
    ")


# Plots  indicating  general  model  fit
for(plotNo in 8:8){
  plot(model2, which=plotNo)
}

# Forecast results
numeric_quantile(27,model2.logdiff.forecast)
mean_error(model2.logdiff.forecast, trueobs)

# Forecasting Plot
plot_forecast_minimized(data=stocks, trueobs=trueobs, n.ahead=27, ylim=400,
              forecasted_model = model2.logdiff.forecast, modeltxt="ARMA-GARCH(1,1)
    ")
```

E-GARCH model:

```
# Project part 2
install.packages("rugarch")
install.packages('Rcpp') # Necessary for bootstrapping

library(rugarch)
library(readxl)
library(Rcpp)
library(forecast)

# Preprocessing and data cleaning
data = read_excel('CBLTCUSD_updated.xls')
data <- na.omit(data)
stocks <- data$CBLTCUSD # Remove dates and convert to numeric
plot(data)

log.stocks <- log(stocks)
diff.log.stocks <- diff(log.stocks)
arima_model_e <- auto.arima(diff.log.stocks)
summary(arima_model_e) #Confirms that ARMA(1,1) is the best mean model for our
    initial series

# Testing observations
trueobs_data <- read_excel('CBLTCUSD_update_2.xls')
trueobs <- trueobs_data$CBLTCUSD


# Helper functions

# Naive/Brute force p and q selection
naive_selection <- function(data, pqmax, distrmod){
  # Helper to select p and q based on lowest AIC/BIC
  # :param data:      should be stocks or the log-diff transformed stock data
  # :param pqmax:     range of combinations to examine, e.g. pqmax=3 will check
  #                   every combination between (0,0) and (3,3)
  # :param distrmod:  Distribution model, e.g. "std" or "norm"
  # :return:          Array of lowest AIC/BIC (p and q printed to console)
  info.res <- vector(mode="list", length=4)
  names(info.res) <- c("p", "q", "aic", "bic")
  pb_iter <- (pqmax+1)^2 # Number of iterations
  pb_count <- 0
  pb <- txtProgressBar(min=pb_count, max=pb_iter, style=3, width=50, char="=")

  for(p in 0:pqmax){
    for(q in 0:pqmax){
      # Create and fit model for each (p,q)
      tmp_model <- ugarchfit(
        spec=ugarchspec(
```

```
          variance.model = list(garchOrder=c(p,q)),
          mean.model = list(armaOrder=c(1,1), include.mean = FALSE),
          distribution.model = distrmod
        ),
        data=data
      )
      # log(log(nObs)) may produce NaNs in some cases, so
      # we skip iterations with invalid AIC/BIC
      skip_iter <- FALSE
      tryCatch(
        calc <- infocriteria(tmp_model),
        error = function(e){ skip_iter <<- TRUE}
      )
      if(skip_iter) { next }

      info.res$p <- append(info.res$p, p)
      info.res$q <- append(info.res$q, q)

      # Extract first 2 results, since we just want Akaike and Bayes
      info.res$aic <- append(info.res$aic, calc[1])
      info.res$bic <- append(info.res$bic, calc[2])
      # Update progress bar
      pb_count <- pb_count + 1
      setTxtProgressBar(pb, pb_count)
    }
  }
  print("Completed naive p, q test")
  info.optimal <- vector(mode="list", length=4)
  names(info.optimal) <- c("aic_val", "bic_val", "aic_param", "bic_param")

  best_aic_index = match(min(info.res$aic), info.res$aic)
  best_bic_index = match(min(info.res$bic), info.res$bic)

  info.optimal$aic_val <- append(info.optimal$aic_val, min(info.res$aic))
  info.optimal$bic_val <- append(info.optimal$bic_val, min(info.res$bic))

  info.optimal$aic_param <- append(
    info.optimal$aic_param,
    c(info.res$p[best_aic_index], info.res$q[best_aic_index])
  )
  info.optimal$bic_param <- append(
    info.optimal$bic_param,
    c(info.res$p[best_bic_index], info.res$q[best_bic_index])
  )

  cat("Best AIC for (p,q) = (", info.optimal$aic_param[1],
      ",", info.optimal$aic_param[2], ")", sep="")
  cat("Best BIC for (p,q) = (", info.optimal$bic_param[1],
      ",", info.optimal$bic_param[2], ")", sep="")
  return(info.optimal)
}

best_choice_ged <- naive_selection(diff.log.stocks, 4, "ged")



# Backtransform log-diff ts
init = stocks[1]
difflogdata = diff.log.stocks
backtransform <- function(x, difflogdata, init){
  n <- length(difflogdata)
  ts.all <- c(difflogdata,x)
  cs <- cumsum(ts.all)
  newts <- exp(cs[(n+1):length(ts.all)])*init
  return(newts)
}

# Plot forecasted values
plot_forecast <- function(data, trueobs, n.ahead, ylim, forecasted_model, modeltxt)
    {
  # Generalized plotter function to plot all forecasts similarly
  # data = time series used in modeling
  # trueobs = testing data not used in modeling
```

```
  # n.ahead = number of time steps forecasted for
  # forecasted_model = Bootrap-forecasted and backtransformed data
  # modeltxt = string representation of model used

  # Plot forecast of backtransformed data
  n <- length(data)
  plot(1:n,stocks[1:n],type="l",xlim=c(as.integer(n/1.25),n+n.ahead),ylim=c(0,ylim)
    , xlab = "Timeframe", ylab = "Value of stock",
       main= paste("Forecasting Litecoin using", modeltxt, "model"))
  for(i in 1:nrow(forecasted_model)){
    lines(seq(n+1,n+n.ahead),forecasted_model[i,],col="lightgrey")
  }
  lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, mean),col="red")
  lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.025),col="
    blue")
  lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.975),col="
    blue")
  lines(seq(n+1,n+n.ahead), trueobs,col="darkgreen")
  legend("topleft", legend=c(
    "Previous stock value",
    "Mean forecast",
    "97.5% confidence interval",
    "Actual observation"),
    col=c("black", "red", "blue", "darkgreen"),
    lty=1, cex=1.05)
}

plot_forecast_minimized <- function(data, trueobs, n.ahead, ylim, forecasted_model,
     modeltxt){
  # Generalized plotter function to plot all forecasts similarly
  # data = time series used in modeling
  # trueobs = testing data not used in modeling
  # n.ahead = number of time steps forecasted for
  # forecasted_model = Bootrap-forecasted and backtransformed data
  # modeltxt = string representation of model used

  # Plot forecast of backtransformed data
  n <- length(data)
  plot(n+1, 200, xlim=c(n+1,n+n.ahead),ylim=c(100,ylim),
       xlab="Timeframe", ylab="Value",
       main= paste("Forecast, ", modeltxt, "model"))
  for(i in 1:nrow(forecasted_model)){
    lines(seq(n+1,n+n.ahead),forecasted_model[i,],col="lightgrey")
  }
  lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, mean),col="red")
  lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.025),col="
    blue")
  lines(seq(n+1,n+n.ahead),apply(forecasted_model, 2, quantile,probs=0.975),col="
    blue")
  lines(seq(n+1,n+n.ahead), trueobs,col="darkgreen")
}

# >>>>> Model 0 : eGARCH(1,1)
best_choice_norm <- naive_selection(diff.log.stocks, 15, "ged") #Confirms that (p,q
    ) = (1,1) is the overall best fit

model0.logdiff <- ugarchfit(
  spec=ugarchspec(
    variance.model = list(model = "eGARCH",garchOrder=best_choice_ged$aic_param),
    mean.model = list(armaOrder=c(1,1)),
    distribution.model = "ged"
  ),
  data=diff.log.stocks
)

# Including the News Impact Curve
for(plotNo in 8:12){
  plot(model0.logdiff, which=plotNo)
}

model0.logdiff.forecast <- ugarchboot(model0.logdiff, n.ahead=27,
                                      method="Partial", n.bootpred = 100000)
model0.logdiff.forecast <- t(
```

```
  apply(model0.logdiff.forecast @ fseries,
        1,
        backtransform,
        difflogdata=difflogdata,
        init=init)
)
plot_forecast(data=stocks, trueobs=trueobs, n.ahead=27, ylim=400,
              forecasted_model = model0.logdiff.forecast, modeltxt="EGARCH(1,1)") # Full plot
plot_forecast_minimized(data=stocks, trueobs=trueobs, n.ahead=27, ylim=400,
              forecasted_model = model0.logdiff.forecast, modeltxt="EGARCH(1,1)") # Only forecasted values

# Calculating the mean prediction error and the confidence intervals
mean_pred <- matrix(ncol = 27, nrow = 1)

for (i in 1:27){
  mean_pred[1,i] <- mean(model0.logdiff.forecast[,i], na.rm = TRUE)
```

Bibliography

[1] Peter Brockwell and Richard Davis. *An Introduction to Time Series and Forecasting*. Vol. 39. Jan. 2002. ISBN: 978-1-4757-2528-5. DOI: 10.1007/978-1-4757-2526-1.

[2] Marthea Elva Ramirez Guzman. *GARCH - Modelling Conditional Variance & Useful Diagnostic Tests*. 2020. URL: https://logicalerrors.wordpress.com/2017/08/14/garch-modeling-conditional-variance-useful-diagnostic-tests/.

[3] Lie Martin Loisel Camille Nilsen Frederick. *Forecasting a cryptocurrency*. 2021. URL: https://github.com/Remi9-l/Timeseries/blob/main/Time_Series_Project_1.pdf.